Efficiency of Tissue P Systems with Cell Separation

Linqiang Pan[1,2], Mario J. Pérez-Jiménez[3]

[1] Key Laboratory of Image Processing and Intelligent Control
Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, People's Republic of China
lqpan@mail.hust.edu.cn
[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, 41012 Sevilla, Spain
marper@us.es

Summary. The most investigated variants of P systems in the last years are cell-like models, especially in terms of efficiency. Recently, different new models of tissue-like (symport/antiport) P systems have received important attention. This paper presents a new class of tissue P systems with cell separation, where cell separation can generate new workspace. Its efficiency is investigated, specifically, (a) only tractable problem can be efficiently solved by using cell separation and communication rules with length at most 1, and (b) an efficient (uniform) solution to SAT problem by using cell separation and communication rules with length at most 6 is presented. Further research topics and open problems are discussed, too.

1 Introduction

Membrane computing is inspired by the structure and the function of living cells, as well as from the organization of cells in tissues, organs, and other higher order structures. The devices of this model, called *P systems*, provide distributed parallel and non-deterministic computing models.

Roughly speaking, the main components of such a model are a cell-like *membrane structures*, in the *compartments* of which one places *multisets* of *symbol-objects* which evolve in a synchronous maximally parallel manner according to given *evolution rules*, also associated with the membranes (for introduction see [19] and for further bibliography see [29]).

Membrane computing is a young branch of natural computing initiated by Gh. Păun in the end of 1998 [17]. It has received important attention from the scientific community since then, with contributions by computer scientists, biologists,

formal linguists and complexity theoreticians, enriching each others with results, open problems and promising new research lines. In fact, membrane computing was selected by the Institute for Scientific Information, USA, as a fast *Emerging Research Front* in computer science, and [20] was mentioned in [28] as a highly cited paper in October 2003.

In the last years, many different models of P systems have been proposed. The most studied variants are characterized by a *cell-like* membrane structure, where the communication takes place between a membrane and the surrounding one. In this model we have a set of nested membranes, in such a way that the graph of neighborhood relation is a tree.

One of the topics in the field is the study of the computational power and efficiency of P systems. In particular, different models of these cell-like P systems have been successfully used in order to design solutions to **NP**-complete problems in polynomial time (see [6] and the references therein). These solutions are obtained by generating an exponential amount of workspace in polynomial time and using parallelism to check simultaneously all the candidate solutions. Inspired by living cell, several ways for obtaining exponential workspace in polynomial time were proposed: membrane division (*mitosis*) [18], membrane creation (*autopoiesis*) [8] membrane separation (*membrane fission*) [16]. These three ways have given rise to the corresponding P systems model: *P systems with active membranes*, *P systems with membrane creation*, and *P systems with membranes separation*. These three models are universal from a computational point of view, but technically, they are pretty different. In fact, nowadays there does not exist any theoretical result which proves that these models can simulate each other in polynomial time.

Under the hypothesis $\mathbf{P}\neq\mathbf{NP}$, Zandron et al. [27] established the limitations of P systems that do not use membrane division concerning the efficient solution of **NP**-complete problems. This result was generalized by Pérez-Jiménez et al. [24] obtaining a characterization of the $\mathbf{P}\neq\mathbf{NP}$ conjecture by the polynomial time unsolvability of an **NP**-complete problem by means of language accepting P systems (without using rules that allow to increase the size of the structure of membranes).

Here, we shall focus on another type of P systems, the so-called (because of their membrane structure) *tissue P systems*. Instead of considering a hierarchical arrangement, membranes are placed in the nodes of a virtual graph. This variant has two biological inspirations (see [15]): intercellular communication and cooperation between neurons. The common mathematical model of these two mechanisms is a net of processors dealing with symbols and communicating these symbols along channels specified in advance. The communication among cells is based on symport/antiport rules, which were introduced to P systems in [20]. Symport rules move objects across a membrane together in one direction, whereas antiport rules move objects across a membrane in opposite directions.

From the seminal definitions of tissue P systems [14, 15], several research lines have been developed and other variants have arisen (see, for example, [1, 2, 3, 9, 10, 26]). One of the most interesting variants of tissue P systems was presented in [22]. In that paper, the definition of tissue P systems is combined with the one of P

systems with active membranes, yielding *tissue P systems with cell division*, and a polynomial-time uniform solution to the **NP**-complete problem SAT is shown. In this kind of tissue P systems [22], there exists replication, that is, the two new cells generated by a division rule have exactly the same objects except for at most a pair of different objects. However, in the biological phenomenon of separation, the contents of the two new cells evolved from a cell can be significantly different, and membrane separation inspired by this biological phenomenon in the framework of cell-like P systems was proved to be an efficient way to obtain exponential workspace in polynomial time [16]. In this paper, a new class of tissue P systems based on cell separation, called *tissue P systems with cell separation*, is presented, and a linear time uniform solution to the **NP**-complete problem SAT is shown.

The paper is organized as follows: first, we recall some preliminaries, and then, the definition of tissue P systems with cell separation is given. Next, recognizer tissue P systems are briefly described. In Section 5, we prove that only tractable problem can be efficiently solved by using cell separation and communication rules with length at most 1. In Section 6, an efficient (uniform) solution to SAT problem by using cell separation and communication rules with length at most 6 is shown, including a short overview of the computation and of the necessary resources. The formal verification of the solution is also given. Finally, some discussion is presented.

2 Preliminaries

An *alphabet*, Σ, is a non empty set, whose elements are called *symbols*. An ordered sequence of symbols is a *string*. The number of symbols in a string u is the *length* of the string, and it is denoted by $|u|$. As usual, the empty string (with length 0) will be denoted by λ. The set of strings of length n built with symbols from the alphabet Σ is denoted by Σ^n and $\Sigma^* = \cup_{n\geq 0}\Sigma^n$. A *language* over Σ is a subset from Σ^*.

A *multiset* m over a set A is a pair (A, f) where $f : A \to \mathbb{N}$ is a mapping. If $m = (A, f)$ is a multiset then its *support* is defined as $supp(m) = \{x \in A \mid f(x) > 0\}$ and its *size* is defined as $\sum_{x\in A} f(x)$. A multiset is empty (resp. finite) if its support is the empty set (resp. finite).

If $m = (A, f)$ is a finite multiset over A, and $supp(m) = \{a_1, \ldots, a_k\}$, then it will be denoted as $m = \{\{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}\}$. That is, superscripts indicate the multiplicity of each element, and if $f(x) = 0$ for any $x \in A$, then this element is omitted.

In what follows we assume the reader is already familiar with the basic notions and the terminology of P systems. For details, see [19].

3 Tissue P Systems with Cell Separation

In the first definition of the model of tissue P systems [14, 15], the membrane structure did not change along the computation. We will give a new model of *tissue P systems with cell separation* based on the cell-like model of P systems with membranes separation [16]. The biological inspiration is clear: alive tissues are not *static* network of cells, since new cells are generated by membrane fission in a natural way.

The main features of this model, from the computational point of view, are that cells are not polarized (the contrary holds in the cell-like model of P systems with active membranes, see [19]); the cells obtained by separation have the same labels as the original cell and if a cell is separated, its interaction with other cells or with the environment is blocked during the separation process. In some sense, this means that while a cell is separating it closes its communication channels.

Formally, a *tissue P system with cell separation* of degree $q \geq 1$ is a tuple

$$\Pi = (\Gamma, O_1, O_2, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_0),$$

where:

1. Γ is a finite *alphabet* whose elements are called *objects*, $\Gamma = O_1 \cup O_2$, $O_1, O_2 \neq \emptyset$, $O_1 \cap O_2 = \emptyset$;
2. $w_1, \ldots, w_q$ are strings over Γ, representing the multisets of objects placed in the q cells of the system at the beginning of the computation;
3. $\mathcal{E} \subseteq \Gamma$ is a finite alphabet representing the set of objects in the environment in arbitrary copies each;
4. $\mathcal{R}$ is a finite set of rules of the following forms:
   (a) $(i, u/v, j)$, for $i, j \in \{0, 1, 2, \ldots, q\}, i \neq j$, $u, v \in \Gamma^*$;
   Communication rules; $1, 2, \cdots, q$ identify the cells of the system, 0 is the environment; when applying a rule $(i, u/v, j)$, the objects of the multiset represented by u are sent from region i to region j and simultaneously the objects of the multiset v are sent from region j to region i, ($|u| + |v|$ is called the length of the communication rule $(i, u/v, j)$);
   (b) $[a]_i \rightarrow [O_1]_i[O_2]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a \in \Gamma$;
   Separation rules; in reaction with an object a, the cell is separated into two cells with the same label; at the same time, object a is consumed; the objects from O_1 are placed in the first cell, those from O_2 are placed in the second cell;
5. $i_0 \in \{0, 1, 2, \ldots, q\}$ is the output cell.

The rules of a system like the above one are used in the non-deterministic maximally parallel manner as customary in membrane computing. At each step, all cells which can evolve must evolve in a maximally parallel way (in each step we apply a multiset of rules which is maximal, no further rule can be added). This way of applying rules has only one restriction: when a cell is separated, the separation rule is the only one which is applied for that cell in that step; the objects

inside that cell do not evolve by means of communication rules. The new cells could participate in the interaction with other cells or the environment by means of communication rules in the next step – providing that they are not separated once again. Their labels precisely identify the rules which can be applied to them.

The configuration of tissue P system with cell separation is described by all multisets of objects associated with all the cells and environment $(w'_1, \cdots, w'_q; w'_0)$, where w'_0 is a multiset over $\Gamma - \mathcal{E}$ representing the objects present in the environment having a finite multiplicity. The tuple $(w_1, w_2, \cdots, w_q; \emptyset)$ is the initial configuration. The computation starts from the initial configuration and proceeds as defined above; only halting computations give a result, and the result is encoded by the objects present in cell i_0 in the halting configuration.

4 Recognizer Tissue P Systems with Cell Separation

NP-completeness has been usually studied in the framework of *decision problems*. Let us recall that a decision problem is a pair (I_X, θ_X) where I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total boolean function over I_X.

In order to study the computing efficiency, the notions from classical *computational complexity theory* are adapted for membrane computing, and a special class of cell-like P systems is introduced in [25]: *recognizer P systems*. For tissue P systems, with the same idea as recognizer cell-like P systems, *recognizer tissue P systems* is introduced in [22].

A *recognizer tissue P system with cell separation* of degree $q \geq 1$ is a construct

$$\Pi = (\Gamma, O_1, O_2, \Sigma, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_{in}, i_0)$$

where:

- $(\Gamma, O_1, O_2, w_1, \ldots, w_q, \mathcal{E}, \mathcal{R}, i_o)$ is a tissue P system with cell separation of degree $q \geq 1$ (as defined in the previous section).
- The working alphabet Γ has two distinguished objects `yes` and `no`, at least one copy of them present in some initial multisets w_1, ..., w_q, but none of them are present in $\mathcal{E}$.
- Σ is an (input) alphabet strictly contained in Γ.
- $i_{in} \in \{1, \ldots, q\}$ is the input cell.
- The output region i_0 is the environment.
- All computations halt.
- If $\mathcal{C}$ is a computation of Π, then either object `yes` or object `no` (but not both) must have been released into the environment, and only at the last step of the computation.

The computations of the system Π with input $w \in \Sigma^*$ start from a configuration of the form $(w_1, w_2, \ldots, w_{i_{in}} w, \ldots, w_q; \emptyset)$, that is, after adding the multiset w to the contents of the input cell i_{in}. We say that $\mathcal{C}$ is an accepting computation

(respectively, rejecting computation) if object `yes` (respectively, `no`) appears in the environment associated to the corresponding halting configuration of $\mathcal{C}$.

We denote by $\mathbf{TSC}(k)$ the class of recognizer tissue P systems with cell separation and with communication rules of length at most k.

Definition 1. *We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer tissue P systems with cell separation if the following holds:*

- *The family $\mathbf{\Pi}$ is* polynomially uniform *by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ from $n \in \mathbb{N}$.*
- *There exists a pair (cod, s) of polynomial-time computable functions over I_X such that:*
  - *for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$;*
  - *the family $\mathbf{\Pi}$ is* polynomially bounded *with regard to (X, cod, s), that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $cod(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps;*
  - *the family $\mathbf{\Pi}$ is* sound *with regard to (X, cod, s), that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))$ with input $cod(u)$, then $\theta_X(u) = 1$;*
  - *the family $\mathbf{\Pi}$ is* complete *with regard to (X, cod, s), that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u))$ with input $cod(u)$ is an accepting one.*

We denote by $\mathbf{PMC}_{\mathbf{TSC}(k)}$ the set of all decision problems which can be solved by means of recognizer tissue P systems from $\mathbf{TSC}(k)$.

5 Limitation on the Efficiency of TSC(1)

In this section, we consider the efficiency of tissue P systems with cell separation and communication rules with length 1. Specifically, we shall prove that such systems can efficiently solve only tractable problems.

Let Π be a tissue P system with cell separation and let all communication rules be of length 1. In this case, each rule of the system can be activated by a single object. Hence, there exists, in some sense, a *dependency* between the object triggering the rule and the object or objects produced by its application. This dependency allows to adapt the ideas developed in [13] for cell-like P systems with active membranes to tissue P systems with cell separation and communication rules of length 1.

We can consider a general pattern $(a, i) \rightarrow (b_1, j) \dots (b_s, j)$ where $i, j \in \{0, 1, 2, \dots, q\}$, and $a, b_k \in \Gamma$, $k \in \{1, \cdots, s\}$. This pattern can be interpreted

as follows: from the object a in the cell (or in the environment) labeled with i we can *reach* the objects $b_1, \ldots, b_s$ in the cell (or in the environment) labeled with j. Communication rules correspond to the case $s = 1$ and $b_1 = a$.

Without loss of generality we can assume that all communication rules in the system obey the syntax $(i, a/\lambda, j)$, since every rule of the form $(j, \lambda/a, i)$ can be rewritten to follow the above syntax, with equivalent semantics.

We formalize these ideas in the following definition.

Definition 2. *Let $\Pi = (\Gamma, \Sigma, \Omega, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in})$ be a tissue P system of degree $q \geq 1$ with cell separation. Let $H = \{0, 1, \ldots, q\}$. The dependency graph associated with Π is the directed graph $G_\Pi = (V_\Pi, E_\Pi)$ defined as follows:*
$V_\Pi = \{(a, i) \in \Gamma \times H : \exists j \in H \ ((i, a/\lambda, j) \in R \ \vee \ (j, a/\lambda, i) \in R)\}$,
$E_\Pi = \{((a, i), (b, j)) : (a = b \ \wedge \ (i, a/\lambda, j) \in R)\}$.

Note that when a separation rule is applied, objects do not evolve, and the labels of membranes do not change, so separation rules do not add any nodes and edges to the associated dependency graph.

Proposition 1. *Let $\Pi = (\Gamma, \Sigma, \Omega, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in})$ be a tissue P system with cell separation, in which the length of all communication rules is 1. Let $H = \{0, 1, \ldots, q\}$. There exists a deterministic Turing machine that constructs the dependency graph G_Π associated with Π, in polynomial time (that is, in a time bounded by a polynomial function depending on the total number of communication rules).*

Proof. A deterministic algorithm that, given a P system Π with the set R_c of communication rules, constructs the corresponding dependency graph, is the following:

```
   Input: Π (with R_c as its set of communication rules)
V_Π ← ∅; E_Π ← ∅
for each rule r ∈ R_c of Π do
  if r = (i, a/λ, j) then
    V_Π ← V_Π ∪ {(a, i), (a, j)}; E_Π ← E_Π ∪ {((a, i), (a, j))}
```

The running time of this algorithm is bounded by $O(|R_c|)$.

Proposition 2. *Let $\Pi = (\Gamma, \Sigma, \Omega, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in})$ be a tissue P system with cell separation, in which the length of all communication rules is 1. Let $H = \{0, 1, \ldots, q\}$. Let Δ_Π be defined as follows:*

$$\Delta_\Pi = \{(a, i) \in \Gamma \times H : \textit{there exists a path (within the dependency graph) from } (a, i) \textit{ to } (\mathtt{yes}, 0)\}.$$

Then, there exists a Turing machine that constructs the set Δ_Π in polynomial time (that is, in a time bounded by a polynomial function depending on the total number of communication rules).

Proof. We can construct the set Δ_Π from Π as follows:

- We construct the dependency graph G_Π associated with Π.
- Then we consider the following algorithm:

```
Input: G_Π = (V_Π, E_Π)
  Δ_Π ← ∅
  for each (a, i) ∈ V_Π do
    if reachability (G_Π, (a, i), (yes, 0)) = yes then
      Δ_Π ← Δ_Π ∪ {(a, i)}
```

The running time of this algorithm [3] is of order $O(|V_\Pi| \cdot |V_\Pi|^2)$, hence it is of order $O(|\Gamma|^3 \cdot |H|^3)$.

Notation: Let $\Pi = (\Gamma, \Sigma, \Omega, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$ be a tissue P system with cell separation. Let m be a multiset over Σ. Then we denote $\mathcal{M}_j^* = \{(a, j) : a \in \mathcal{M}_j\}$, for $1 \leq j \leq q$, and $m^* = \{(a, i_{in}) : a \in m\}$.

Below we characterize accepting computations of a recognizer tissue P system with cell separation and communication rules of length 1 by distinguished paths in the associated dependency graph.

Lemma 1. *Let $\Pi = (\Gamma, \Sigma, \Omega, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in})$ be a recognizer confluent tissue P system with cell separation in which the length of all communication rules is 1. The following assertions are equivalent:*

(1) There exists an accepting computation of Π.
(2) There exists $(a_0, i_0) \in \bigcup_{j=1}^{q} \mathcal{M}_j^*$ and a path in the dependency graph associated with Π, from (a_0, i_0) to $(\mathtt{yes}, 0)$.

Proof. (1) $\Rightarrow$ (2). First, we show that for each accepting computation $\mathcal{C}$ of Π there exists $(a_0, i_0) \in \bigcup_{j=1}^{q} \mathcal{M}_j^*$ and a path $\gamma_\mathcal{C}$ in the dependency graph associated with Π from (a_0, i_0) to $(\mathtt{yes}, 0)$. By induction on the length n of $\mathcal{C}$.

If $n = 1$, a single step is performed in $\mathcal{C}$ from C_0 to C_1. A rule of the form $(j, \mathtt{yes}/\lambda, 0)$, with $\mathtt{yes} \in \Gamma, j \neq 0$, has been applied in that step. Then, $(\mathtt{yes}, j) \in \mathcal{M}_j^*$, for some $j \in \{1, \ldots, q\}$. Hence, $((\mathtt{yes}, j), (\mathtt{yes}, 0))$ is a path in the dependency graph associated with Π.

Let us suppose that the result holds for n. Let $\mathcal{C} = (C_0, C_1, \ldots, C_n, C_{n+1})$ be an accepting computation of Π. Then $\mathcal{C}' = (C_1, \ldots, C_n, C_{n+1})$ is an accepting computation of the system $\Pi' = (\Gamma, \Sigma, \Omega, \mathcal{M}_1', \ldots, \mathcal{M}_q', \mathcal{R}, i_{in})$, where $\mathcal{M}_j'$ is the

[3] The Reachability Problem is the following: *given a (directed or undirected) graph, G, and two nodes a, b, determine whether or not the node b is reachable from a, that is, whether or not there exists a path in the graph from a to b.* It is easy to design an algorithm running in polynomial time solving this problem. For example, given a (directed or undirected) graph, G, and two nodes a, b, we consider a depth–first–search with source a, and we check if b is in the tree of the computation forest whose root is a. The total running time of this algorithm is $O(|V| + |E|)$, that is, in the worst case is quadratic in the number of nodes. Moreover, this algorithm needs to store a linear number of items (it can be proved that there exists another polynomial time algorithm which uses $O(\log^2(|V|))$ space).

content of cell j in configuration C_1, for $1 \leq j \leq q$. By induction hypothesis there exists an object b_0 in a cell i_0 from C_1, and a path in the dependency graph associated with Π' from (b_0, i_0) to $(\texttt{yes}, 0)$. If (b_0, i_0) is an element of configuration C_0 (that means that in the first step a separation rule has been applied to cell i_0), then the result holds. Otherwise, there is an element (a_0, j_0) in C_0 producing (b_0, i_0). So, there exists a path $\gamma_{\mathcal{C}}$ in the dependency graph associated with Π from (a_0, j_0) to $(\texttt{yes}, 0)$.

(2) $\Rightarrow$ (1). Let us prove that for each $(a_0, i_0) \in \bigcup_{j=1}^{q} \mathcal{M}_j^*$ and for each path in the dependency graph associated with Π from (a_0, i_0) to $(\texttt{yes}, 0)$, there exists an accepting computation of Π. By induction on the length n of the path.

If $n = 1$, we have a path $((a_0, i_0), (\texttt{yes}, 0))$. Then, $a_0 = \texttt{yes}$ and the computation $\mathcal{C} = (C_0, C_1)$ where the rule $(i_0, \texttt{yes}/\lambda, 0)$ belongs to a multiset of rules m_0 that produces configuration C_1 from C_0 is an accepting computation of Π.

Let us suppose that the result holds for n. Let

$$((a_0, i_0), (a_1, i_1), \ldots (a_n, i_n), (\texttt{yes}, 0))$$

be a path in the dependency graph of length $n + 1$. Let C_1 be the configuration of Π reached from C_0 by the application of a multiset of rules containing the rule that produces (a_1, i_1) from (a_0, i_0). Then $((a_1, i_1), \ldots (a_n, i_n), (\texttt{yes}, 0))$ is a path of length n in the dependency graph associated with the system

$$\Pi' = (\Gamma, \Sigma, \Omega, \mathcal{M}_1', \ldots, \mathcal{M}_q', \mathcal{R}, i_{in})$$

where $\mathcal{M}_j'$ is the content of cell j in configuration C_1, for $1 \leq j \leq q$. By induction hypothesis, there exists an accepting computation $\mathcal{C}' = (C_1, \ldots, C_t)$ of Π'. Hence, $\mathcal{C} = (C_0, C_1, \ldots, C_t)$ is an accepting computation of Π.

Next, given a family $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ of recognizer tissue P system with cell separation in which the length of all communication rules is 1, solving a decision problem, we will characterize the acceptance of an instance of the problem, w, using the set $\Delta_{\Pi(s(w))}$ associated with the system $\Pi(s(w))$, that processes the given instance w. More precisely, the instance is accepted by the system if and only if there is an object in the initial configuration of the system $\Pi(s(w))$ with input $cod(w)$ such that there exists a path in the associated dependency graph starting from that object and reaching the object $\texttt{yes}$ in the environment.

Proposition 3. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ be a family of recognizer tissue P system with cell separation in which the length of all communication rules is 1 solving X, according to Definition 1. Let (cod, s) be the polynomial encoding associated with that solution. Then, for each instance w of the problem X the following assertions are equivalent:*

(a) $\theta_X(w) = 1$ (that is, the answer to the problem is $\texttt{yes}$ for w).

(b) $\Delta_{\Pi(s(w))} \cap ((cod(w))^* \cup \bigcup_{j=1}^{p} \mathcal{M}_j^*) \neq \emptyset$, where $\mathcal{M}_1, \ldots, \mathcal{M}_p$ are the initial multisets of the system $\Pi(s(w))$.

Proof. Let $w \in I_X$. Then $\theta_X(w) = 1$ if and only if there exists an accepting computation of the system $\Pi(s(w))$ with input multiset $cod(w)$. By Lemma 1, this condition is equivalent to the following: in the initial configuration of $\Pi(s(w))$ with input multiset $cod(w)$ there exists at least one object $a \in \Gamma$ in a cell labelled with i such that in the dependency graph the node $(\texttt{yes}, 0)$ is reachable from (a, i).

Hence, $\theta_X(w) = 1$ if and only if $\Delta_{\Pi(s(w))} \cap \mathcal{M}_j^* \neq \emptyset$ for some $j \in \{1, \ldots, p\}$, or $\Delta_{\Pi(s(w))} \cap (cod(w))^* \neq \emptyset$.

Theorem 1. $\mathbf{P} = \boldsymbol{PMC}_{TSC(1)}$

Proof. We have $\mathbf{P} \subseteq \mathbf{PMC}_{TSC(1)}$ because the class $\mathbf{PMC}_{TSC(1)}$ is closed under polynomial time reduction. In what follows, we show that $\mathbf{PMC}_{TSC(1)} \subseteq \mathbf{P}$. Let $X \in \mathbf{PMC}_{TSC(1)}$ and let $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ be a family of recognizer tissue P systems with cell separation solving X, according to Definition 1. Let (cod, s) be the polynomial encoding associated with that solution.

We consider the following deterministic algorithm:

```
Input: An instance w of X
- Construct the system Π(s(w)) with input multiset cod(w).
- Construct the dependency graph G_{Π(s(w))} associated with Π(s(w)).
- Construct the set Δ_{Π(s(w))} as indicated in Proposition 2
  answer ← no; j ← 1
  while j ≤ p ∧ answer = no do
      if Δ_{Π(s(w))} ∩ M_j^* ≠ ∅ then
          answer ← yes
      j ← j + 1
  endwhile
  if Δ_{Π(s(w))} ∩ (cod(w))^* ≠ ∅ then
      answer ← yes
```

On one hand, the answer of this algorithm is `yes` if and only if there exists a pair (a, i) belonging to $\Delta_{\Pi(s(w))}$ such that the symbol a appears in the cell labelled with i in the initial configuration (with input the multiset $cod(w)$).

On the other hand, a pair (a, i) belongs to $\Delta_{\Pi(s(w))}$ if and only if there exists a path from (a, i) to $(\texttt{yes}, 0)$, that is, if and only if we can obtain an accepting computation of $\Pi(s(w))$ with input $cod(w)$. Hence, the algorithm above described solves the problem X.

The cost to determine whether or not $\Delta_{\Pi(s(w))} \cap \mathcal{M}_j^* \neq \emptyset$ (or $\Delta_{\Pi(s(w))} \cap (cod(w))^* \neq \emptyset$) is of order $O(|\Gamma|^2 \cdot |H|^2)$.

Hence, the running time of this algorithm can be bounded by $f(|w|) + O(|R_c|) + O(q \cdot |\Gamma|^2 \cdot n^2)$, where f is the (total) cost of a polynomial encoding from X to $\mathbf{\Pi}$, R_c is the set of rules of $\Pi(s(w))$, and q is the number of (initial) cells of $\Pi(s(w))$. Furthermore, by Definition 1 we have that all involved parameters are polynomial in $|w|$. That is, the algorithm is polynomial in the size $|w|$ of the input.

6 Solving Computationally Hard Problems by Using TSC(6)

In this section, we consider the efficiency of tissue P systems with cell separation and communication rules of length at most 6. As expected, such systems can efficiently solve computationally hard problems. In what follows, we show how to efficiently solve SAT problem by such systems.

The SAT problem is the following: given a boolean formula in conjunctive normal form (CNF), to determine whether or not there exists an assignment to its variables on which it evaluates true. This is a well known **NP**-complete problem [5].

The solution proposed follows a brute force approach in the framework of recognizer P systems with cell separation. The solution consists of the following stages:

- *Generation Stage*: All truth assignments for the n variables are produced by using cell separation in an adequate way.
- *Checking Stage*: We determine whether there is a truth assignment that makes the boolean formula evaluate to true.
- *Output Stage*: The system sends to the environment the right answer according to the results of the previous stage.

Let us consider the polynomial time computable function $\langle n, m\rangle = ((m + n)(m + n + 1)/2) + m$ (the pair function), which is a primitive recursive and bijective function from $\mathbb{N}^2$ to $\mathbb{N}$. We shall construct a family $\mathbf{\Pi} = \{\Pi(t) \mid t \in \mathbb{N}\}$ such that each system $\Pi(t)$ will solve all instances of SAT problem with n variables and m clauses, where $t = \langle n, m\rangle$, provided that the appropriate input multisets are given.

For each $n, m \in \mathbb{N}$,

$$\Pi(\langle n, m\rangle) = (\Gamma(\langle n, m\rangle), \Sigma(\langle n, m\rangle), w_1, w_2, \mathcal{R}(\langle n, m\rangle), \mathcal{E}(\langle n, m\rangle), i_{in}, i_0),$$

with the following components:

- $\Gamma(\langle n, m\rangle) = O_1 \cup O_2$,
  $O_1 = \{x_{i,j}, \bar{x}_{i,j}, c_{i,j}, z_{i,j} \mid 1 \le i \le n, 1 \le j \le m\} \cup$
  $\{A_i \mid 1 \le i \le n\} \cup \{a_{1,i}, b_{1,i}, g_i, h_i \mid 1 \le i \le n-1\} \cup$
  $\{d_{1,i}, e_i, l_i \mid 1 \le i \le n-2\} \cup \{a_{2,i}, b_{2,i}, d_{2,i} \mid 2 \le i \le n-1\} \cup$
  $\{a_{i,j,k}, b_{i,j,k}, d_{i,j,k} \mid 1 \le i \le n, 1 \le j \le m, 1 \le k \le n-1\} \cup$
  $\{B_i \mid 1 \le i \le 4n\} \cup \{C_i \mid 1 \le i \le 3n\} \cup$
  $\{D_i \mid 1 \le i \le 4n + 2m\} \cup \{E_i \mid 1 \le i \le 4n + 2m + 3\} \cup$
  $\{r_j \mid 1 \le j \le m\} \cup \{T_i, F_i, t_i, f_i \mid 1 \le i \le n\} \cup \{c, p, s, y, z, \texttt{yes}, \texttt{no}\}$,
  $O_2 = \{x'_{i,j}, \bar{x}'_{i,j}, z'_{i,j}\} \mid 1 \le i \le n, 1 \le j \le m\} \cup$
  $\{T'_i, F'_i \mid 1 \le i \le n\} \cup \{y', z'\}$.
- $\Sigma(\langle n, m\rangle) = \{c_{i,j}, x_{i,j}, \bar{x}_{i,j} \mid 1 \le i \le n, 1 \le j \le m\}$.
- $w_1 = \{\{a_{1,1}, a_{2,2}, g_1, B_1, C_1, D_1, E_1, p, \texttt{yes}, \texttt{no}\}\} \cup$
  $\{\{a_{i,j,1} \mid 1 \le i \le n, 1 \le j \le m\}\}$.

- $w_2 = A_1$.
- $\mathcal{R}(\langle n, m\rangle)$ is the set of rules:
  1. **Separation rule:**
     $r_1 \equiv [s]_2 \rightarrow [O_1]_2[O_2]_2$.
  2. **Communication rules:**
     $r_{2,i,j,k} \equiv (1, a_{i,j,k}/b_{i,j,k}, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq k \leq n-1$;
     $r_{3,i,j,k} \equiv (1, b_{i,j,k}/c_{i,j}^2 d_{i,j,k}^2, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq k \leq n-2$;
     $r_{4,i,j} \equiv (1, b_{i,j,n-1}/c_{i,j}^2, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
     $r_{5,i,j,k} \equiv (1, d_{i,j,k}/a_{i,j,k+1}, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq k \leq n-2$;
     $r_{6,i} \equiv (1, a_{1,i}/b_{1,i}, 0)$ for $1 \leq i \leq n-1$;
     $r_{7,i} \equiv (1, b_{1,i}/c^2 d_{1,i}^2 e_i^2, 0)$ for $1 \leq i \leq n-2$;
     $r_8 \equiv (1, b_{1,n-1}/c^2, 0)$;
     $r_{9,i} \equiv (1, d_{1,i}/a_{1,i+1}, 0)$ for $1 \leq i \leq n-2$;
     $r_{10,i} \equiv (1, e_i/a_{2,i+1}, 0)$ for $1 \leq i \leq n-2$;
     $r_{11,i} \equiv (1, a_{2,i}/b_{2,i}, 0)$ for $2 \leq i \leq n-1$;
     $r_{12,i} \equiv (1, b_{2,i}/c^2 d_{2,i}^2, 0)$ for $2 \leq i \leq n-2$;
     $r_{13} \equiv (1, b_{2,n-1}/c^2, 0)$;
     $r_{14,i} \equiv (1, d_{2,i}/a_{2,i+1}, 0)$ for $2 \leq i \leq n-2$;
     $r_{15,i} \equiv (1, g_i/h_i, 0)$ for $1 \leq i \leq n-1$;
     $r_{16,i} \equiv (1, h_i/l_i^2 A_{i+1}^2, 0)$ for $1 \leq i \leq n-2$;
     $r_{17} \equiv (1, h_{n-1}/A_n^2, 0)$;
     $r_{18,i} \equiv (1, l_i/g_{i+1}, 0)$ for $1 \leq i \leq n-2$;
     $r_{19,i,j} \equiv (2, c_{i,j}x_{i,j}/z_{i,j}z'_{i,j}x_{i,j}x'_{i,j}, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
     $r_{20,i,j} \equiv (2, c_{i,j}\bar{x}_{i,j}/z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
     $r_{21,i,j} \equiv (2, c_{i,j}x'_{i,j}/z_{i,j}z'_{i,j}x_{i,j}x'_{i,j}, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
     $r_{22,i,j} \equiv (2, c_{i,j}\bar{x}'_{i,j}/z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
     $r_{23,i} \equiv (2, A_i/T_i F'_i zz'yy's, 0)$ for $1 \leq i \leq n-1$;
     $r_{24} \equiv (2, A_n/T_n F'_n yy's, 0)$;
     $r_{25,i} \equiv (2, cT_i/zz'T_iT'_i, 0)$ for $1 \leq i \leq n-1$;
     $r_{26,i} \equiv (2, cT'_i/zz'T_iT'_i, 0)$ for $1 \leq i \leq n-1$;
     $r_{27,i} \equiv (2, cF_i/zz'F_iF'_i, 0)$ for $1 \leq i \leq n-1$;
     $r_{28,i} \equiv (2, cF'_i/zz'F_iF'_i, 0)$ for $1 \leq i \leq n-1$;
     $r_{29,i,j} \equiv (1, c_{i,j}/z_{i,j}, 2)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
     $r_{30,i,j} \equiv (1, c_{i,j}/z'_{i,j}, 2)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
     $r_{31} \equiv (1, c/z, 2)$;
     $r_{32} \equiv (1, c/z', 2)$;
     $r_{33,i} \equiv (1, A_i/y, 2)$ for $2 \leq i \leq n$;
     $r_{34,i} \equiv (1, A_i/y', 2)$ for $2 \leq i \leq n$;
     $r_{35,i} \equiv (1, B_i/B_{i+1}, 0)$ for $1 \leq i \leq 2n-1$;
     $r_{36,i} \equiv (1, B_i/B_{i+1}^2, 0)$ for $2n \leq i \leq 3n-1$;
     $r_{37,i} \equiv (1, C_i/C_{i+1}, 0)$ for $1 \leq i \leq 2n-1$;
     $r_{38,i} \equiv (1, C_i/C_{i+1}^2, 0)$ for $2n \leq i \leq 3n-1$;
     $r_{39,i} \equiv (1, D_i/D_{i+1}, 0)$ for $1 \leq i \leq 2n-1$;
     $r_{40,i} \equiv (1, D_i/D_{i+1}^2, 0)$ for $2n \leq i \leq 3n-1$;

$r_{41,i} \equiv (1, E_i/E_{i+1}, 0)$ for $1 \leq i \leq 4n + 2m + 2$;
$r_{42,i,j} \equiv (1, z_{i,j}/\lambda, 0)$;
$r_{43,i,j} \equiv (1, z'_{i,j}/\lambda, 0)$;
$r_{44} \equiv (1, y/\lambda, 0)$;
$r_{45} \equiv (1, y'/\lambda, 0)$;
$r_{46} \equiv (1, z/\lambda, 0)$;
$r_{47} \equiv (1, z'/\lambda, 0)$;
$r_{48} \equiv (1, B_{3n}C_{3n}D_{3n}/y, 2)$;
$r_{49} \equiv (1, B_{3n}C_{3n}D_{3n}/y', 2)$;
$r_{50,i} \equiv (2, C_{3n}T_i/C_{3n}t_i, 0)$ for $1 \leq i \leq n$;
$r_{51,i} \equiv (2, C_{3n}T'_i/C_{3n}t_i, 0)$ for $1 \leq i \leq n$;
$r_{52,i} \equiv (2, C_{3n}F_i/C_{3n}f_i, 0)$ for $1 \leq i \leq n$;
$r_{53,i} \equiv (2, C_{3n}F'_i/C_{3n}f_i, 0)$ for $1 \leq i \leq n$;
$r_{54,i} \equiv (2, B_i/B^2_{i+1}, 0)$ for $3n \leq i \leq 4n - 1$;
$r_{55,i} \equiv (2, D_i/D_{i+1}, 0)$ for $3n \leq i \leq 4n - 1$;
$r_{56,i,j} \equiv (2, B_{4n}t_i x_{i,j}/B_{4n}t_i r_j, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{57,i,j} \equiv (2, B_{4n}t_i \bar{x}_{i,j}/B_{4n}t_i, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{58,i,j} \equiv (2, B_{4n}t_i x'_{i,j}/B_{4n}t_i r_j, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{59,i,j} \equiv (2, B_{4n}t_i \bar{x}'_{i,j}/B_{4n}t_i, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{60,i,j} \equiv (2, B_{4n}f_i \bar{x}_{i,j}/B_{4n}f_i r_j, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{61,i,j} \equiv (2, B_{4n}f_i x_{i,j}/B_{4n}f_i, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{62,i,j} \equiv (2, B_{4n}f_i \bar{x}'_{i,j}/B_{4n}f_i r_j, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{63,i,j} \equiv (2, B_{4n}f_i x'_{i,j}/B_{4n}r_j, 0)$ for $1 \leq i \leq n$, $1 \leq j \leq m$;
$r_{64,i} \equiv (2, D_i/D_{i+1}, 0)$ for $4n \leq i \leq 4n + m - 1$;
$r_{65,i} \equiv (2, D_{4n+m+i}r_{i+1}/D_{4n+m+i+1}, 0)$ for $0 \leq i \leq m - 1$;
$r_{66} \equiv (2, D_{4n+2m}/\lambda, 1)$;
$r_{67} \equiv (1, D_{4n+2m}\ p\ \texttt{yes}/\lambda, 0)$;
$r_{68} \equiv (1, E_{4n+2m+3}\ p\ \texttt{no}/\lambda, 0)$.

- $\mathcal{E}(\langle n, m\rangle) = \Gamma(\langle n, m\rangle) - \{\texttt{yes}, \texttt{no}\}$.
- $i_{in} = 2$ is the *input cell*.
- $i_0 = 0$ is the *output region*.

6.1 An Overview of the Computation

A family of recognizer tissue P systems with cell separation is constructed above. For an instance of SAT problem $\varphi = M_1 \wedge \cdots \wedge M_m$, consisting of m clauses $M_i = y_{i,1} \vee \cdots \vee y_{i,l_i}$, $1 \leq i \leq m$, where $Var(\varphi) = \{x_1, \cdots, x_n\}$, $y_{i,k} \in \{x_j, \neg x_j \mid 1 \leq j \leq n\}$, $1 \leq i \leq m, 1 \leq k \leq l_i$, the *size* mapping on the set of instances is defined as $s(\varphi) = \langle n, m\rangle = ((m + n)(m + n + 1)/2) + m$, the encoding of the instance is the multiset $cod(\varphi) = \{\{c_{i,j}x_{i,j} \mid x_i \in \{y_{j,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq n, 1 \leq j \leq m\}\} \cup \{\{c_{i,j}\bar{x}_{i,j} \mid \neg x_i \in \{y_{j,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq n, 1 \leq j \leq m\}\}$.

Now, we informally describe how system $\Pi(s(\varphi))$ with input $cod(\varphi)$ works.

Let us start with the *generation stage*. This stage has several parallel processes, which we describe in several items.

- In cells with label 2, by rules $r_{19,i,j}$ – $r_{22,i,j}$, objects $c_{i,j}x_{i,j}$, $c_{i,j}\bar{x}_{i,j}$, $c_{i,j}x'_{i,j}$, $c_{i,j}\bar{x}'_{i,j}$ introduce objects $z_{i,j}z'_{i,j}x_{i,j}x'_{i,j}$, $z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}$, $z_{i,j}z'_{i,j}x_{i,j}x'_{i,j}$, $z_{i,j}z'_{i,j}\bar{x}_{i,j}\bar{x}'_{i,j}$, respectively. In the next step, the objects with prime and the objects without prime are separated into the new daughter cells with label 2. The idea is that $c_{i,j}$ is used to duplicate $x_{i,j}$ and $\bar{x}_{i,j}$ (in the sense ignoring the prime), so that one copy of each of them will appear in each cell with label 2. The objects $z_{i,j}$ and $z'_{i,j}$ in cells with label 2 are exchanged with the objects $c_{i,j}$ in the cell with label 1 by the rules $r_{29,i,j}$ and $r_{30,i,j}$. In this way, the cycle of duplication-separation can be iterated.

- In parallel with the above duplication-separation process, the objects c are used to duplicate the objects T_i, T'_i, F_i, and F'_i by the rules $r_{25,i}$ – $r_{28,i}$; the rules r_{31} and r_{32} take care of introducing the object c from the cell with label 1 to cells with label 2.

- In the initial configuration of the system, the cell with label 2 contains an object A_1 (A_i encodes the i-th variable in the propositional formula). The objects T_1, F'_1, z, z', y, y' and s are brought in the cell with label 2, in exchange of A_1, by the rule $r_{23,i}$. The objects T_1 and F'_1 correspond to the values *true* and *false* which the variable x_1 may assume (in general, T_i (or T'_i) and F_i (or F'_i) are for the variable x_i), and in the next step they are separated into the new daughter cells with label 2 by separation rule, because $T_1 \in O_1$ and $F'_1 \in O_2$. The object s is used to activate the separation rule r_1, and is consumed during the application of separation rule. The objects y and y' are used to introduce A_2 from the cell with label 1, and the process of truth-assignment for variable x_2 can continue. In this way, in $3n-1$ steps, we get 2^n cells with label 2, and each one contains one of the 2^n possible truth-assignments for the n variables.

- In parallel with the operations in the cells with label 2, the objects $a_{i,j,k+1}$ from the cell with label 1 are traded for objects $b_{i,j,k+1}$ from the environment at the step $3k+1$ ($0 \leq k \leq n-3$) by the rule $r_{2,i,j,k}$. In the next step, each object $b_{i,j,k+1}$ is traded for two copies of objects $c_{i,j}$ and $d_{i,j,k+1}$ by the rule $r_{3,i,j,k}$. At step $3k+3$ ($0 \leq k \leq n-3$), the object $d_{i,j,k}$ is traded for object $a_{i,j,k+2}$ by the rule $r_{4,i,j,k}$. Especially, at step $3n-5$, $a_{i,j,n-1}$ is traded for $b_{i,j,n-1}$ by the $r_{2,i,j,k}$, at step $3n-4$, each copy of object $b_{i,j,n-1}$ is traded for two copies of $c_{i,j}$ by the $r_{4,i,j}$. After step $3n-4$, there is no object $a_{i,j,k}$ appears in the cell with label 1, and the group of rules $r_{2,i,j,k}$ – $r_{5,i,j,k}$ will not be used again. Note that the subscript k of the object $a_{i,j,k}$ grows by 1 in every 3 steps until reaching the value $n-1$, and the number of copies of $a_{i,j,k}$ is doubled in every 3 steps. At step $3k+3$ ($0 \leq k \leq n-2$), the cell with label 1 has 2^{k+1} copies of object $c_{i,j}$. At the same time, we have 2^{k+1} cells with label 2, and each cell with label 2 contains one copy of object $z_{i,j}$ or one copy of object $z'_{i,j}$. Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of $c_{i,j}$ from the cell with label 1 by the rules $r_{29,i,j}$ and $r_{30,i,j}$. The object $c_{i,j}$ in cell with label 2 is used for duplication as described above.

- The objects $a_{1,i}$ and $a_{2,i}$ in the cell with label 1 has the similar role as object $a_{i,j,k}$ in cell 1, which introduce appropriate copies of object c for the duplication of objects T_i, T'_i, F_i, and F'_i by the rules $r_{6,i} - r_{14,i}$. Note that at step $3k+3$ ($0 \leq k \leq n-2$), there are $2^{k+1}(k+1)$ copies of object c, which ensure each cell with label 2 gets $k+1$ copies of object c by the maximality of the parallelism of using the rules.
- The object g_{i+1} in the cell with label 1 is traded for h_{i+1} from the environment at step $3i+1$ ($0 \leq i \leq n-3$) by the rule $r_{15,i}$. In the next step, the object h_{i+1} is traded for two copies of objects l_{i+1} and A_{i+2} by the rule $r_{13,i}$. At the step $3i+3$ ($0 \leq i \leq n-3$), the object l_{i+1} is traded for two copies of g_{i+2}, so that the process can be iterated, until the subscript i of g_i reaches $n-1$. Especially, at step $3n-5$, object g_{n-1} is traded for h_{n-1} by the rule $r_{15,i}$, at step $3n-4$, each object h_{n-1} is traded for two copies of A_n. After step $3n-4$, there is no object g_i appears in the cell with label 1, and the group of rules $r_{15,i} - r_{18,i}$ will not be used again. At the step $3i+3$ ($0 \leq i \leq n-2$), the cell with label 1 contains 2^{i+1} copies of A_{i+2}, and we have 2^{i+1} cells with label 2, each of them contains one copy of object y or one copy of object y'. Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of A_{i+2} from the cell 1 by the rules $r_{33,i}$ and $r_{34,i}$. In this way, the truth-assignment for the valuable x_{i+1} can continue.
- The counters B_i, C_i, D_i, and E_i in the cell with label 1 grow their subscripts by the rules $r_{35,i} - r_{41,i}$. From step $2n$ to step $3n-1$, the number of copies of objects of the first three types is doubled, hence after $3n-1$ steps, the cell with label 1 contains 2^n copies of B_{3n}, C_{3n}, and D_{3n}. Objects B_i will check which clauses are satisfied by a given truth-assignment, objects C_i are used to multiply the number of copies of t_i, f_i as we will see immediately, objects D_i are used to check whether there is at least one truth-assignment which satisfies all clauses, and E_i will be used to bring the object `no` to the environment, if this will be the case, in the end of the computation.
- The objects $z_{i,j}$, $z'_{i,j}$, y, y', z, and z' in the cell with label 1 are removed by the rules $r_{42,i,j} - r_{47}$. (Actually, if the objects $z_{i,j}$, $z'_{i,j}$, y, y', z, and z' stay in the cell with label 1, they do not influence the work of the system. The rules $r_{38} - r_{43}$ are designed just in order to simplify the formal verification.)

In this way, after the $(3n-1)$–th step the generation stage finishes and the *checking stage* starts. At this moment, the cell with label 1 contains 2^n copies of objects B_{3n}, C_{3n}, and D_{3n}, and there are 2^n cells with label 2, each of them containing a copy of y and $n-1$ copies of z, or a copy of y' and $n-1$ copies of z'. The objects z and z' in cells with label 2 will not evolve anymore, because the cell with label 1 contains no object c from now on, and the rules r_{31} and r_{32} can not be applied.

At the step $3n$, objects y or y' are traded for objects B_{3n}, C_{3n}, and D_{3n} by rules r_{48} and r_{49}. (Note that the rules $r_{33,i}$ and $r_{34,i}$ can not be used, because there is no object A_i in the cell with label 1 at this moment and henceforth. And

the cells with label 2 cannot separate any more.) Due to the maximality of the parallelism of using the rules, each cell with label 2 gets exactly one copy of each of B_{3n}, C_{3n}, and D_{3n}.

In the presence of C_{3n}, the objects T_i and T'_i, F_i and F'_i introduce the objects t_i and f_i, respectively. We have only one copy of C_{3n} available, for each one of t_i and f_i we need one step. So this phase needs n steps that is, this phase ends at step $4n$.

In parallel with the previous operations, the counters B_i and D_i increase their subscripts, until reaching the value $4n$ by the rules $r_{54,i}$ and $r_{55,i}$. Each cell with label 2 contains one copy of D_{4n} and 2^n copies of B_{4n}. Simultaneously, E_i increase its subscript in the cell with label 1.

At step $4n+1$, with the presence of B_{4n}, we start to check the values assumed by clauses for the truth-assignments from each cell with label 2 by the rules $r_{56,i,j}$ – $r_{63,i,j}$. Each membrane with label 2 contains nm objects $x_{i,j}$ and $\bar{x}_{i,j}$ or nm objects $x'_{i,j}$ and $\bar{x}'_{i,j}$, because each clause contains at most n literals, and we have m clauses. Note that each membrane with label 2 contains 2^n copies of B_{4n} and n objects t_i and f_i. At each step, n objects $x_{i,j}$ and $\bar{x}_{i,j}$, or n objects $x'_{i,j}$ and $\bar{x}'_{i,j}$ are checked. So it takes m steps. In parallel, D_i increases the subscript, until reaching the value $4n+m$ (at step $4n+m$) by the rule $r_{64,i}$.

By the rule $r_{65,i}$, in each cell with label 2, we check whether or not all clauses are satisfied by the corresponding truth-assignment. For each clause which is satisfied, we increase by one the subscript of D_i, hence the subscript reaches the value $4n+2m$ if and only if all clauses are satisfied.

The output stage starts at the $(4n+2m+1)$-th step.

- *Affirmative answer*: If one of the truth-assignments from a cell with label 2 has satisfied all clauses, then in that cell there is an object D_{4n+2m} as described above, which is sent to the cell with label 1 by the rule r_{66}. In the next step, the object `yes` leaves the system by the rule r_{67}, signaling the fact that the formula is satisfiable. In cell 1, the counter E_i increases its subscript by the rule $r_{41,i}$, until reaching the value $4n+2m+3$, but after that it will remain unchanged – it can leave the cell with label 1 only in the presence of p, but this object p was already moved to the environment at step $4n+2m+2$. The computation halts at step $4n+2m+2$.
- *Negative answer*: If the counter E_i reaches the subscript $4n+2m+3$ and the object p is still in the cell with label 1, then the object `no` can be moved to the environment by the rule r_{68}, signaling that the formula is not satisfiable. The computation finishes at step $4n+2m+3$.

6.2 Formal Verification

In this subsection, we prove that the family built above solves SAT problem in a polynomial time, according to Definition 1. First of all, the Definition 1 requires that the defined family is *consistent*, in the sense that all systems of the family

must be recognizer tissue P systems with cell separation. By the construction (type of rules and working alphabet) it is clear that it is a family of tissue P systems with cell separation. In order to show that all members in $\mathbf{\Pi}$ are recognizer systems it suffices to check that all the computations halt (this will be deduced from the polynomial bound), and that either an object `yes` or an object `no` is sent out exactly in the last step of the computation (this will be deduced from the soundness and completeness).

Polynomial uniformity of the family

We will show that the family $\mathbf{\Pi} = \{\Pi(\langle n, m\rangle) \mid n, m \in \mathbb{N}\}$ defined above is polynomially uniform by Turing machines. To this aim it will be proved that $\Pi(\langle n, m\rangle)$ is built in polynomial time with respect to the size parameter n and m of instances of SAT problem.

It is easy to check that the rules of a system $\Pi(\langle n, m\rangle)$ of the family are defined recursively from the values n and m. And the necessary resources to build an element of the family are of a polynomial order, as shown below:

- Size of the alphabet: $3n^2m + 4nm + 30n + 5m - 5 \in O(n^2m)$.
- Initial number of cells: $2 \in O(1)$.
- Initial number of objects: $nm + 10 \in O(nm)$.
- Number of rules: $3n^2m + 15nm + 36n + 3m - 12 \in O(n^2m)$.
- Maximal length of a rule: $6 \in O(1)$.

Therefore, a deterministic Turing machine can build $\Pi(\langle n, m\rangle)$ in a polynomial time with respect to n and m.

Polynomial bound of the family

For an instance of SAT problem $\varphi = M_1 \wedge \cdots \wedge M_m$, consisting of m clauses $M_i = y_{i,1} \vee \cdots \vee y_{i,l_i},\ 1 \leq i \leq m$, where $Var(\varphi) = \{x_1, \cdots, x_n\}$, $y_{i,k} \in \{x_j, \neg x_j \mid 1 \leq j \leq n\},\ 1 \leq i \leq m, 1 \leq k \leq l_i$, we recall the *size* mapping function $s(\varphi)$ and the encoding function $cod(\varphi)$: $s(\varphi) = \langle n, m\rangle$, and $cod(\varphi) = \{\{c_{i,j}x_{i,j} \mid x_i \in \{y_{j,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq n, 1 \leq j \leq m\}\} \cup \{\{c_{i,j}\bar{x}_{i,j} \mid \neg x_i \in \{y_{j,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq n, 1 \leq j \leq m\}\}$. The pair (cod, s) is computable in polynomial time, $cod(\varphi)$ is an input multiset of the system $\Pi(s(\varphi))$.

In order to prove that the system $\Pi(s(\varphi))$ with input $cod(\varphi)$ is polynomially bounded, it suffices to find the moment in which the computation halts, or at least, an upper bound for it.

Proposition 4. *The family* $\mathbf{\Pi} = \{\Pi(\langle n, m\rangle) \mid n, m \in \mathbb{N}\}$ *is polynomially bounded with respect to* (SAT, cod, s).

Proof. We will informally go through the stages of the computation in order to estimate a bound for the number of steps. The computation will be checked more in detail when addressing the soundness and completeness proof.

Let $\varphi = M_1 \wedge \cdots \wedge M_m$ be an instance of the problem SAT. We shall study what happens during the computation of the system $\Pi(s(\varphi))$ with input $cod(\varphi)$ in order to find the halting step, or at least, an upper bound for it.

First, the generation stage has exactly $3n-1$ steps, where at steps $3k+2$ $(0 \leq k \leq n-1)$ the cells with label 2 are separated. In this way, we get 2^n cells with label 2, each of them contains one of the 2^n possible truth-assignments for the n variables.

After one more step, the objects B_{3n}, C_{3n}, and D_{3n} arrive at cells with label 2, and the checking stage starts. The object C_{3n} works for n steps introducing objects t_i or f_i into cells with label 2, until all object T_i, T_i', F_i and F_i' are consumed, at the step $4n$. From step $4n+1$, the objects B_{4n} start to work checking which clauses are satisfied by the truth-assignment from each cell with label 2. This checking takes m steps. When the subscript of D_i grows to $4n+m$ at step $4n+m$, the system starts to check whether or not all clauses are satisfied by the corresponding truth-assignment. It takes m steps, and the checking stage ends at step $4n+2m$.

The last one is the answer stage. The longest case is obtained when the answer is negative. In this case there are two steps where only the counter E_i is working. At the step $4n+2m+3$ the object $E_{4n+2m+3}$ works together with object p bringing `no` from the cell with label 1 into the environment.

Therefore, there exists a linear bound (with respect to n and m) on the number of steps of the computation.

Soundness and completeness of the family

In order to prove the soundness and completeness of the family $\mathbf{\Pi}$ with respect to (SAT, cod, s), we shall prove that for a given instance φ of the problem SAT, the system $\Pi(s(\varphi))$ with input $cod(\varphi)$ sends out an object `yes` if and only if the answer to the problem for the considered instance φ is affirmative and the object `no` is sent out otherwise. In both cases the answer will be sent to the environment in the last step of the computation.

For the sake of simplicity in the notation, we consider the following two functions $\psi(\sigma_j(x_i))$ and $\gamma(\sigma_j(x_i))$. Let $\mathcal{F}$ be the set of all assignments of the variables $x_1, x_2, \cdots, x_n$. We order the set $\mathcal{S}$ in lexicographical order, that is, $\mathcal{F} = \{\sigma_1, \sigma_2, \cdots, \sigma_{2^n}\}$, where $\sigma_j(x_i) \in \{0,1\}$ $(1 \leq j \leq 2^n,\ 1 \leq i \leq n)$ is an assignment of variables. For $1 \leq j \leq 2^n$, $1 \leq i \leq n$, we define ψ as follows: if j is odd, then

$$\psi(\sigma_j(x_i)) = \begin{cases} T_i, & \text{if } \sigma_j(x_i) = 1, \\ F_i, & \text{if } \sigma_j(x_i) = 0; \end{cases}$$

if j is even, then

$$\psi(\sigma_j(x_i)) = \begin{cases} T_i', & \text{if } \sigma_j(x_i) = 1, \\ F_i', & \text{if } \sigma_j(x_i) = 0. \end{cases}$$

For each assignment $\sigma_j(x_i) \in \{0,1\}$, and for $i = 1, \ldots, n$, we define γ as follows:

$$\gamma(\sigma_j(x_i)) = \begin{cases} t_i, & \text{if } \sigma_j(x_i) = 1; \\ f_i, & \text{if } \sigma_j(x_i) = 0. \end{cases}$$

In this way, each assignment of variables σ_j is associated a multiset $\{\{\psi(\sigma_j(x_1)), \psi(\sigma_j(x_2)), \cdots, \psi(\sigma_j(x_n))\}\}$ and a multiset $\{\{\gamma(\sigma_j(x_1)), \gamma(\sigma_j(x_2)), \cdots, \gamma(\sigma_j(x_n))\}\}$.

Given a computation $\mathcal{C}$ we denote the configuration at the i-th step as $\mathcal{C}_i$. Moreover, $\mathcal{C}_i(1)$ will denote the multiset associated to cell 1 in such configuration.

Proposition 5. *Let $\mathcal{C}$ be an arbitrary computation of the system, then at step $3k+2$ for all k such that $0 \leq k \leq n-2$, the cell with label 1 gets 2^{k+1} copies of object $c_{i,j}$, $2^{k+1}(k+1)$ copies of object c, and 2^{k+1} copies of object A_{k+2} from the environment. And after step $3n-3$, the cell with label 1 cannot get objects $c_{i,j}$, c, A_i any more.*

Proof. It is not difficult to find that in the set of all rules there are 6 types of rules related to object $c_{i,j}$, that is, rules $r_{2,i,j,k}$ – $r_{5,i,j,k}$, $r_{29,i,j}$ and $r_{30,i,j}$. The rules $r_{29,i,j}$ and $r_{30,i,j}$ are used to move $c_{i,j}$ from the cell with label 1 to cells with label 2 in exchange of $z_{i,j}$ or $z'_{i,j}$, which happens at steps $3k+3$ for all k such that $0 \leq k \leq n-2$. Anyway, these two rules do not bring object $c_{i,j}$ into the cell with label 1. So we need only to check how these 4 types of rules $r_{2,i,j,k}$ – $r_{5,i,j,k}$ work.

First, by induction on k, we prove that at step $3k+2$ ($0 \leq k \leq n-3$), the cell with label 1 gets 2^{k+1} copies of object $c_{i,j}$ and the cell with label 1 has exactly 2^{k+1} copies of $d_{i,j,k+1}$.

In the multiset $\mathcal{C}_0(1)$, there is one copy of each object $a_{i,j,1}$ ($1 \leq i \leq n$, $1 \leq j \leq m$). By application of rules $r_{2,i,j,1}$ and $r_{3,i,j,1}$, the cell with label 1 gets two 2 copies of $c_{i,j}$ and 2 copies $d_{i,j,1}$ at step 2.

Now suppose the result is true for $k < n-4$. We have, by the inductive hypothesis, at step $3k+2$, the cell with label 1 gets 2^{k+1} copies of object $c_{i,j}$ and the cell with label 1 has exactly 2^{k+1} copies of $d_{i,j,k+1}$. At step $3k+3$, among these 4 types of rules $r_{2,i,j,k+1}$ – $r_{5,i,j,k}$, only rule $r_{5,i,j,k+1}$ can be applied, 2^{k+1} copies of $d_{i,j,k+1}$ are traded for 2^{k+1} copies of $a_{i,j,k+2}$. At step $3(k+1)+1$, 2^{k+1} copies of $a_{i,j,k+2}$ are traded for 2^{k+1} copies of $b_{i,j,k+2}$ by the $r_{2,i,j,k+2}$. At step $3(k+1)+2$, by the rule $r_{3,i,j,k+2}$, the cell with label 1 gets 2^{k+2} copies of object $c_{i,j}$ and the cell with label 1 has exactly 2^{k+2} copies of $d_{i,j,k+2}$.

Based on the above result, specifically, we have, at step $3(n-3)+2$, the cell with label 1 has exactly 2^{n-2} copies of $d_{i,j,n-2}$. In the next 3 steps, the rules $r_{5,i,j,n-2}$, $r_{2,i,j,n-1}$, and $r_{4,i,j,n-1}$ are applied in order. At step $3(n-2)+2$, the cell with label 1 gets 2^{n-1} copies of $c_{i,j}$. Note that on object $d_{i,j,k}$ is brought into the cell with label 1 at step $3(n-2)+2$, and the group of rules $r_{5,i,j,k}$, $r_{2,i,j,k}$, and $r_{4,i,j,k}$ cannot be used any more. Therefore the cell with label 1 will not get object $c_{i,j}$ any more, and the result holds.

For the cases of objects c and A_i, the results can be proved similarly. We omit them here.

Proposition 6. *Let $\mathcal{C}$ be an arbitrary computation of the system, then*

1. *For each assignment $\sigma_j(x_i)$, $i = 1, 2, \cdots, n$, there exists only one cell with label 2 in $\mathcal{C}_{3n-1}$ that contains multiset $\{\{\psi(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$.*

2. *There exist exactly* 2^n *cells with label* 2 *in configuration* $\mathcal{C}_k$ *(*$k \geq 3n-1$*). Particularly, in configuration* $\mathcal{C}_{3n-1}$*, each cell with label* 2 *contains a multiset* $\{\{y\}\} \cup \{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ *or a multiset* $\{\{y'\}\} \cup \{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$.

Proof. We prove the result by induction.

In the configuration $\mathcal{C}_0$, there is only one cell with label 2, which has multiset $\{\{c_{ij}, x_{i,j}, \bar{x}_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}\} \cup \{\{A_1\}\}$. The rules $r_{19,i,j} - r_{22,i,j}$ and $r_{23,1}$ can be applied. At step 1, the cell with label 2 has multiset $\{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\} \cup \{\{z, z', T_1, F'_1, y, y', s\}\}$. At step 2, with the appearance of object s, the separation rule r_1 is used to separate cell with label 2, object s is consumed, and the multiset $\{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\} \cup \{\{z, T_1, y\}\}$ are placed in one new cell with label 2, the multiset $\{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\} \cup \{\{z', F'_1, y'\}\}$ are placed in another new cell with label 2. We take the cell with label 2 where $\psi(\sigma_j(x_1))$ appears.

By Proposition 5, at step 2, the cell with label 1 has two copies of $c_{i,j}$, two copies of c, two copies of A_2. So, at step 3, the rules $r_{29,i,j}$, $r_{30,i,j}$, $r_{33,2}$ and $r_{34,2}$ can be applied. Due to the maximality of the parallelism of using rules, each cell with label 2 gets exactly one copy of $c_{i,j}$, one copy of c, and one copy of A_2 from the cell with label 1. Object $c_{i,j}$ and c are used for duplication, and A_2 is used to assign truth-values to the valuable x_2. In this way, the next cycle of duplication-separation can continue.

In general, after step $3k+2$ $(0 \leq k \leq n-1)$ (that is, the second step in the $(k+1)$-th cycle of duplication-separation), we take the cell with label 2 where $\{\{\psi(\sigma_j(x_1)), \cdots, \psi(\sigma_j(x_{k+1}))\}\}$ appears. In this way, at step $3n-1$, there exists exactly one cell with label 2 whose multiset is $\{\{\psi(\sigma_j(x_1)), \cdots, \psi(\sigma_j(x_n))\}\}$. (Note the difference of the rule r_{24} and the rules $r_{23,i}$. The rule r_{24} does not bring objects z and z' into cells with label 2 from the environment, and this rule is used at step $3n-2$. So the object z does not appear in the multiset of cell 2 that corresponds to the assignment σ_j.)

From the above proof, it is easy to see that the multiset $\{\{\psi(\sigma_j(x_1)), \cdots, \psi(\sigma_j(x_n))\}\}$ appears only in the corresponding cell with label 2.

In every cycle of duplication-separation, the number of cells with label 2 is doubled. In the $3n-1$ steps, there are n cycles. So there exist exactly 2^n cells with label 2 in configuration $\mathcal{C}_{3n-1}$; and from now on, cells with label 2 will not separate anymore.

In the last cycle of duplication-separation, at step $3n-2$, each of the 2^{n-1} cells with label 2 contains one copy of y and one copy of y' by the rule r_{24}; at step $3n-1$, the 2^{n-1} cells with label 2 are separated by the rule r_1, each of the 2^n new cells gets one copy of object y or one copy of object y'.

Proposition 7. *Let* $\mathcal{C}$ *be an arbitrary computation of the system, then* $\mathcal{C}_{3n-1}(1) = \{\{B_{3n}^{2^n}, C_{3n}^{2^n}, D_{3n}^{2^n}, E_{3n}, p, \texttt{yes}, \texttt{no}\}\}$.

Proof. In order to prove $\mathcal{C}_{3n-1}(1) = \{\{B_{3n}^{2^n}, C_{3n}^{2^n}, D_{3n}^{2^n}, E_{3n}, p, \texttt{yes}, \texttt{no}\}\}$, we will check how all the rules related to the cell 1 work in the first $3n-1$ steps.

- Checking the rules $r_{2,i,j,k} - r_{18,i}$.
  From the proofs of Propositions 5 and 6, we can find that after step $3n-3$, there are no objects $a_{i,j,k}, b_{i,j,k}, c_{i,j}, d_{i,j,k}, a_{1,i}, b_{1,i}, c, d_{1,i}, e_i, a_{2,i}, b_{2,i}, d_{2,i}, g_i, h_i, l_i, A_i$ in the cell with label 1, and the rules $r_{2,i,j,k} - r_{18,i}$ will not bring any more objects into the cell with label 1.

- Checking the rules $r_{35,i} - r_{40,i}$.
  In the first $2n-1$ steps of the computation, by the rules $r_{35,i}$, $r_{37,i}$, and $r_{39,i}$, the subscripts of B_1, C_1, and D_1 grow to $2n$. In the next n steps, by the rules $r_{36,i}$, $r_{38,i}$, and $r_{40,i}$, the subscripts of B_{2n}, C_{2n}, and D_{2n} grow to $3n$, and at every step, the numbers of objects of each type B_i, C_i, and D_i are doubled. So the cell with label 1 has 2^n copies of B_{3n}, 2^n copies of C_{3n}, and 2^n copies of D_{3n} at the step $3n-1$.

- Checking the rule $r_{41,i}$.
  By the rule $r_{41,i}$, the subscript of E_1 grow to $3n$ in the first $3n-1$ steps of the computation. So the cell 1 has the object E_{3n} at step $3n-1$.

- Checking the group of rules $r_{29,i,j} - r_{34,i,j}$ and the group of rules $r_{42,i,j} - r_{47}$.
  In the first $3n-3$ steps, the cell with label 1 has communication with cells with label 2 getting objects $z_{i,j}$, $z'_{i,j}$, z, z', y, y' from cells with label 2, by the rules $r_{29,i,j} - r_{34,i,j}$. In the next step after the objects $z_{i,j}$, $z'_{i,j}$, z, z', y, y' reach the cell 1, they are sent to the environment by the rules $r_{42,i,j} - r_{47}$.

- Checking the group of rules $r_{48} - r_{49}$.
  At the step $3n-1$, the subscript of objects B_i, C_i and D_i grow to $3n$. The rules $r_{48} - r_{49}$ can be applied at step $3n$. But, in the first $3n-1$ steps of the computation, they cannot be applied.

- Checking the rules $r_{66} - r_{68}$.
  In the first steps $3n-1$, there are no object D_{4n+2m} appearing in cells with label 2, and no object $E_{4n+2m+3}$ appearing in the cell with label 1. The rules $r_{66} - r_{68}$ cannot be applied in the first $3n-1$ steps of the computation, so the cell with label 1 has objects $\texttt{yes}$, $\texttt{no}$ and p at the step $3n-1$.

  Therefore, $\mathcal{C}_{3n-1}(1) = \{\{B_{3n}^{2^n}, C_{3n}^{2^n}, D_{3n}^{2^n}, E_{3n}, p, \texttt{yes}, \texttt{no}\}\}$.

Proposition 8. *Let $\mathcal{C}$ be an arbitrary computation of the system, then*

1. $\mathcal{C}_{3n}(1) = \{\{y^{2^{n-1}}, (y')^{2^{n-1}}, E_{3n+1}, p, \texttt{yes}, \texttt{no}\}\}$;
2. *For each assignment σ_j there exists only one cell with label 2 in $\mathcal{C}_{3n}$ that contains multiset $\{\{\psi(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$. In configuration $\mathcal{C}_{3n}$, each cell with label 2 contains a multiset $\{\{B_{3n}, C_{3n}, D_{3n}\}\} \cup \{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ or multiset $\{\{B_{3n}, C_{3n}, D_{3n}\}\} \cup \{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$.*

Proof. The multiset $\mathcal{C}_{3n}(1)$ is obtained from $\mathcal{C}_{3n-1}(1)$ by the application of the rules $r_{41,i}$, r_{48} and r_{49}. The object E_{3n} in $\mathcal{C}_{3n-1}(1)$ increases its subscript by one by the rule $r_{41,i}$, so one copy of E_{3n+1} appears in $\mathcal{C}_{3n}(1)$. By Proposition 7, at step $3n-1$, the cell with label 1 has 2^n copies of B_{3n}, 2^n copies of C_{3n}, and 2^n copies of D_{3n}. By Proposition 6, at step $3n-1$, there exist exactly 2^n cells with label 2, each of them contains one copy of object y or one copy of object y', and the number of cells with label 2 containing object y (resp. y') is 2^{n-1} (resp. 2^{n-1}). At step $3n$, the rules r_{48} and r_{49} can be applied, 2^n copies of objects $B_{3n}C_{3n}D_{3n}$ in the cell with label 1 are traded for y or y' from the cells with label 2. Due to the maximality of the parallelism of using rules, each cell with label 2 gets one copy of objects $B_{3n}C_{3n}D_{3n}$, and the cell with label 1 gets 2^{n-1} copies of object y and 2^{n-1} copies of object y'. Therefore, $\mathcal{C}_{3n}(1) = \{\{y^{2^{n-1}}, (y')^{2^{n-1}}, E_{3n+1}, p, \texttt{yes}, \texttt{no}\}\}$; and for each assignment σ_j there exists only one cell with label 2 in $\mathcal{C}_{3n}$ that contains multiset $\{\{\psi(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$. By Proposition 6 and the above proof, it is not difficult to see that in configuration $\mathcal{C}_{3n}$, each cell with label 2 contains a multiset $\{\{B_{3n}, C_{3n}, D_{3n}\}\} \cup \{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ or a multiset $\{\{B_{3n}, C_{3n}, D_{3n}\}\} \cup \{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$.

Proposition 9. $\mathcal{C}_{4n}(1) = \{\{E_{4n+1}, p, \texttt{yes}, \texttt{no}\}\}$ *holds.*

Proof. By the rules r_{44} and r_{45}, at step $3n+1$, the objects y and y' are moved to the environment. Henceforth the cell with label 1 will not get any more y or y', because the objects y or y' are traded into cells with label 2 from the environment against the object A_n, and no A_i will appear in cells with label 2 after step $3n-2$. By the rule $r_{41,i}$, the subscript of E_{3n+1} grows to E_{4n+1}. These are all the operations related to the cell with label 1 from step $3n+1$ to step $4n$. Therefore, $\mathcal{C}_{4n}(1) = \{\{E_{4n+1}, p, \texttt{yes}, \texttt{no}\}\}$ holds.

Proposition 10. *For each assignment σ_j, there exists only one cell with label 2 in $\mathcal{C}_{4n}$ that contains multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$. In configuration $\mathcal{C}_{4n}$, each cell with label 2 contains a multiset $\{\{B_{4n}^{2^n}, C_{3n}, D_{4n}\}\} \cup \{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ or a multiset $\{\{B_{4n}^{2^n}, C_{3n}, D_{4n}\}\} \cup \{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$.*

Proof. Based on Proposition 8, we prove Proposition 10 holds.

From step $3n+1$ to step $4n$, the objects in multiset $\{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ and multiset $\{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ keep unchanged because no rules can be applied to them.

By the rules $r_{54,i}$ and $r_{55,i}$, the subscripts of objects B_{3n} and D_{3n} in $\mathcal{C}_{3n}$ increase, until reaching the value $4n$ at step $4n$. At each step from step $3n+1$ to step at step $4n$, the number of object B_i is doubled by the rule $r_{54,i}$, so in configuration $\mathcal{C}_{4n}$, there are 2^n copies of B_i.

For an assignment σ_j, we consider the only cell with label 2 in $\mathcal{C}_{3n}$ that contains multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$. From the definition of functions ψ and γ, we can see that T_i, T'_i and t_i correspond to the value *true* which the valuable x_i

assumes; F_i, F_i' and f_i correspond to the value $false$ which the valuable x_i assumes. Both multisets $\{\{\psi(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$ and $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$ are associated with the same assignment σ_j. By the rules $r_{50,i}$ – $r_{53,i}$, with the object C_{3n}, the objects T_i or T_i' introduce the objects t_i, and the objects F_i or F_i' introduce the objects f_i. Because there is only one copy of C_{3n}, it takes n steps to introduce all the objects in multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$.

Therefore Proposition 10 holds.

Proposition 11. $\mathcal{C}_{4n+m}(1) = \{\{E_{4n+m+1}, p, \texttt{yes}, \texttt{no}\}\}$ *holds.*

Proof. By the rule $r_{41,i}$, the subscript of object E_{4n+1} in $\mathcal{C}_{4n}(1)$ grows to the value $4n + m + 1$ at step $4n + m$. The objects $p, \texttt{yes}, \texttt{no}$ keeps unchanged. Therefore, Proposition 11 holds.

Proposition 12. *For each assignment σ_j, there exists only one cell with label* 2 *in $\mathcal{C}_{4n+m}$ that contains multiset $(\cup_{i=1}^{n}\{\{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = t_i$, and $x_{i,j_l} \in cod(\varphi), l = 1, 2, \cdots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\}\}) \cup (\cup_{i=1}^{n}\{\{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = f_i$, and $\bar{x}_{i,j_l} \in cod(\varphi), l = 1, 2, \cdots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\}\})$. In configuration $\mathcal{C}_{4n+m}$, each cell with label* 2 *contains a multiset $\{\{B_{4n}^{2^n}, C_{3n}, D_{4n+m}\}\}$.*

Proof. By Proposition 10, for each assignment σ_j, there exists only one cell with label 2 in $\mathcal{C}_{4n}$ that contains multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$; and each cell with label 2 contains a multiset $\{\{B_{4n}, C_{3n}, D_{4n}\}\} \cup \{\{z_{i,j}, x_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z_{i,j}, \bar{x}_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ or a multiset $\{\{B_{4n}, C_{3n}, D_{4n}\}\} \cup \{\{z'_{i,j}, x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{z'_{i,j}, \bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$. In the following, we consider this unique cell with label 2 that contains multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$.

The objects C_{3n} keep unchanged, and the subscript of D_{4n} reaches $4n + m$ at step $4n + m$ by the rule $r_{64,i}$.

With the presence of B_{4n} in $\mathcal{C}_{4n}$ (not appearing in $\mathcal{C}_i$ $(i < 4n)$), the rules $r_{56,i,j}$ – $r_{63,i,j}$ can be applied. We start to check which clauses are satisfied. If $\sigma_j((x_i)) = t_i$ and $x_{i,j} \in cod(\varphi)$, then rule $r_{56,i,j}$ or $r_{58,i,j}$ is applied, and an object r_j is introduced into the corresponding cell with label 2. If $\sigma_j((x_i)) = t_i$ and $\bar{x}_{i,j} \in cod(\varphi)$, then rule $r_{57,i,j}$ or $r_{59,i,j}$ is applied, and the object $\bar{x}_{i,j}$ or $\bar{x}'_{i,j}$ is removed from the corresponding cell with label 2. Similarly, if $\sigma_j((x_i)) = f_i$ and $\bar{x}_{i,j} \in cod(\varphi)$, then rule $r_{60,i,j}$ or $r_{62,i,j}$ is applied, and an object r_j is introduced into the corresponding cell with label 2. If $\sigma_j((x_i)) = f_i$ and $x_{i,j} \in cod(\varphi)$, then rule $r_{61,i,j}$ or $r_{63,i,j}$ is applied, and the object $x_{i,j}$ or $x'_{i,j}$ is removed from the corresponding cell with label 2. The sizes of both $cod(\varphi)$ and $\{\{x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{\bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$ are nm, and each cell with label 2 contains multiset $cod(\varphi)$ or $\{\{x'_{i,j} \mid x_{i,j} \in cod(\varphi)\}\} \cup \{\{\bar{x}'_{i,j} \mid \bar{x}_{i,j} \in cod(\varphi)\}\}$. We have 2^n copies of B_{4n}, n objects t_i and f_i from the multiset $\{\{\gamma(\sigma_j(x_i)) \mid i = 1, 2, \cdots, n\}\}$, so it takes m steps to check which clauses are satisfied. In total, all the introduced objects r_i form the multiset $(\cup_{i=1}^{n}\{\{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = t_i, \text{and } x_{i,j_l} \in cod(\varphi), l = 1, 2, \cdots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\}\}) \cup (\cup_{i=1}^{n}\{\{r_{j_1} r_{j_2} \cdots r_{j_k} \mid \gamma(\sigma_j(x_i)) = f_i, \text{and } \bar{x}_{i,j_l} \in cod(\varphi), l = 1, 2, \cdots, k, 1 \leq j_1 < j_2 < \cdots < j_k \leq n\}\})$.

Proposition 13. $\mathcal{C}_{4n+2m}(1) = \{\{E_{4n+2m+1}, p, \texttt{yes}, \texttt{no}\}\}$ *holds.*

Proof. By the rule $r_{41,i}$, the subscript of object E_{4n+m+1} in $\mathcal{C}_{4n+m}(1)$ grows to the value $4n + 2m + 1$ at step $4n + 2m$. The objects $p, \texttt{yes}, \texttt{no}$ keeps unchanged. Therefore, Proposition 13 holds.

Proposition 14. *Let $\mathcal{C}$ be an arbitrary computation of the system, then*

- *If σ_j is an assignment that does not satisfy the formula φ, then there exists only one cell with label 2 in $\mathcal{C}_{4n+2m}$ associated with σ_j, and whose associated multiset contains an object $D_{4n+m+\alpha}$, where $0 \leq \alpha < m$ such that the clauses $M_1, \cdots, M_\alpha$ are satisfied by the assignment f, but $M_{\alpha+1}$ is not satisfied by the assignment σ_j.*
- *If σ_j is an assignment that satisfies the formula φ, then there exists only one cell 2 in $\mathcal{C}_{4n+2m}$ associated with σ_j, and whose associated multiset contains one copy of object D_{4n+2m}.*

Proof. From the configuration $\mathcal{C}_{4n+m}$, we start to check whether or not all clauses are satisfied by the corresponding assignment. Such checking is simultaneous in all 2^n cells with label 2.

Let us consider an assignment σ_j. By Proposition 12, with the presence of object D_{4n+m}, the rule $r_{65,i}$ can be applied. The clauses are checked in the order from M_1 to M_m. For each clause which is satisfied (that is, the corresponding object r_i appears), we increase by one the subscript of D_i, hence the subscript of D_i reaches the value $4n + 2m$ if and only if all clauses are satisfied. If the clauses $M_1, \cdots, M_\alpha$ $(0 \leq \alpha < m)$ are satisfied, but $M_{\alpha+1}$ is not satisfied (that is, $r_1, \cdots, r_\alpha$ appear, but $r_{\alpha+1}$ does not appear), then the subscript of D_i can only reach the value $4n + m + \alpha$.

Therefore, Proposition 14 holds.

Proposition 15. *Let $\mathcal{C}$ be an arbitrary computation of the system, Let us suppose that there exists an assignment that satisfies the formula φ. Then*
(a) $\mathcal{C}_{4n+2m+1}(1) = \{\{E_{4n+2m+2}, D^{\beta}_{4n+2m}, p, \texttt{yes}, \texttt{no}\}\}$,
(b) $\mathcal{C}_{4n+2m+2}(1) = \{\{E_{4n+2m+3}, D^{\beta-1}_{4n+2m}, \texttt{no}\}\}$,
where β is the number of assignments that satisfy the formula φ. Furthermore, the object yes *appears in $\mathcal{C}_{4n+2m+2}(0)$.*

Proof. The configuration of item (a) is obtained by the application of rules $r_{41,i}$ and r_{66} to the previous configuration $\mathcal{C}_{4n+2m}$. By the rule $r_{41,i}$, the object $E_{4n+2m+1}$ in $\mathcal{C}_{4n+2m}(1)$ grows by one its subscript at step $4n+2m+1$. By Proposition 14, for each assignment that satisfies the formula φ, there exists exactly one associated cell with label 2 in $\mathcal{C}_{4n+2m}$ whose multiset contains an object D_{4n+2m}. The object D_{4n+2m} is moved to the cell with label 1 by the rule r_{66}. If there are β assignments that satisfy the formula φ, then the cell 1 gets β copies of object D_{4n+2m}.

The configuration of item (b) is obtained by the application of rules $r_{41,i}$ and r_{67} to the previous configuration $\mathcal{C}_{4n+2m+1}(1)$. By the rule $r_{41,i}$, the object $E_{4n+2m+2}$

in $\mathcal{C}_{4n+2m+1}(1)$ grows by one its subscript at step $4n + 2m + 2$. By the rule r_{67}, the object `yes` together with objects D_{4n+2m} and p leaves the system into the environment, signaling the formula φ is satisfiable. The one copy of object p is consumed by the rule r_{67}, so the rule r_{68} cannot be applied. The object `no` cannot exit into the environment.

Proposition 16. *Let $\mathcal{C}$ be an arbitrary computation of the system, Let us suppose that there does not exist any assignment that satisfies the formula φ. Then*
(a) $\mathcal{C}_{4n+2m+1}(1) = \{\{E_{4n+2m+2}, p, \texttt{yes}, \texttt{no}\}\}$,
(b) $\mathcal{C}_{4n+2m+2}(1) = \{\{E_{4n+2m+3}, p, \texttt{yes}, \texttt{no}\}\}$,
(c) $\mathcal{C}_{4n+2m+3}(1) = \{\{\texttt{yes}\}\}$.
Furthermore, the object `no` *appears in* $\mathcal{C}_{4n+2m+3}(0)$.

Proof. If there does not exist any assignment that satisfies the formula φ, by Proposition 14, all cells with label 2 do not contain object D_{4n+2m}. Of course, the cell with label 1 cannot get object D_{4n+2m}.

The configurations of items (a) an (b) are obtained by the application of rules $r_{41,i}$ to the previous configuration $\mathcal{C}_{4n+2m}$.

The configuration of item (c) is obtained by the application of rules r_{68} to the previous configuration.

6.3 Main Results

The system constructed to solution of SAT in Section 6 has communication rules with length at most 6. From the discussion in the previous sections and according to the definition of solvability given in Section 4, we have the following result:

Theorem 2. $SAT \in \mathbf{PMC}_{TSC(6)}$.

Corollary 1. $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{TS(6)}$.

Proof. It suffices to make the following observations: the SAT problem is **NP**-complete, SAT$\in \mathbf{PMC}_{TS(6)}$ and this complexity class is closed under polynomial-time reduction and under complement.

7 Discussion

The efficiency of cell-like P systems for solving **NP**-complete problems has been widely studied. The space-time tradeoff method is used to efficiently solve **NP**-complete problems in the framework of cell-like P systems. Membrane division, membrane creation, and membrane separation are three efficient ways to obtain exponential workspace in polynomial time. Membrane division is introduced into tissue P systems, and a linear time solution for SAT problem by tissue P systems with cell division is given [22]. In this research, membrane separation is introduced into tissue P systems, and a polynomial time solution for SAT problem by tissue

P systems with cell separation and communication rules of lengths at most 6 is presented. We also prove that tissue P systems with cell separation and communication rules of length 1 can only solve tractable problems. Hence, in the framework of recognizer tissue P systems with cell separation, the lengths of the communication rules provide a borderline between efficiency and non-efficiency. Specifically, a frontier is there when we pass from length 1 to length 6. The role of the lengths of communication rules is worth further investigation. That is, what happens if we consider tissue P systems with communication rules of length k, $k \in \{2, 3, 4, 5\}$?

In the framework of tissue P systems, when cell division is used to generate exponential workspace in polynomial time, there is an advantage: all the other objects in the cell are duplicated except the object that activate the cell division operation. But both cell creation and cell separation have no such duplication function. In this sense, the solution for SAT problem presented in this paper gives some hint for answering the following open problem: how to efficiently solve **NP**-complete problems by tissue P systems with cell creation.

Although SAT problem is **NP**-complete (the other **NP** problems can be reduced to SAT problem in polynomial-time), we would like to stress that up to now there does not exist a methodology to compute the reduction process by P systems. The solution to SAT problem by tissue P systems with cell separation can be used as a scheme for designing solutions to other **NP**–complete problems such as the *vertex–cover* problem, the *clique* problem, the *Hamiltonian path* problem, etc.

Recently, a new kind of P system model, called spiking neural P systems, was introduced [7], which has neural-like architectures. It was proved that spiking neural P systems are Turing complete [7]. About the efficiency of spiking neural P systems to solve computationally hard problems, there is an interesting result: an SN P system of polynomial size cannot solve in a deterministic way in a polynomial time an **NP**-complete problem (unless **P=NP**) [11]. Hence, under the assumption that $\mathbf{P} \neq \mathbf{NP}$, efficient solutions to **NP**-complete problems cannot be obtained without introducing features which enhance the efficiency. One of possible features is some ways to exponentially grow the workspace during the computation. Cell division, cell creation and cell separation are candidates to be introduced into spiking neural P systems for exponential workspace generation. Although the architectures of spiking neural P systems are similar to the architectures of tissue P systems, it is not a trivial work to introduce cell division, cell creation and cell separation into spiking neural P systems and give efficient solutions to **NP**-complete problems by new variants of spiking neural P systems.

In general, it is expected that the research of efficiency and complexity on P systems can provide insight on unconventional parallel computing models, and also help us in clarifying the relations between classic complexity classes.

Acknowledgements

The work of L. Pan was supported by National Natural Science Foundation of China (Grant Nos. 60674106, 30870826, 60703047, and 60533010), Program for

New Century Excellent Talents in University (NCET-05-0612), Ph.D. Programs Foundation of Ministry of Education of China (20060487014), Chenguang Program of Wuhan (200750731262), HUST-SRF (2007Z015A), and Natural Science Foundation of Hubei Province (2008CDB113 and 2008CDB180). The work of M.J. Pérez-Jiménez was supported by Project TIN2006-13452 of the Ministerio de Educación y Ciencia of Spain and Project of Excellence with *Investigador de Reconocida Valía*, from Junta de Andalucía, grant P08 – TIC 04200.

References

1. Alhazov, A., Freund, R. and Oswald, M. Tissue P Systems with Antiport Rules ans Small Numbers of Symbols and Cells. *Lecture Notes in Computer Science* **3572**, (2005), 100–111.
2. Bernardini, F. and Gheorghe, M. Cell Communication in Tissue P Systems and Cell Division in Population P Systems. *Soft Computing* **9**, 9, (2005), 640–649.
3. Freund, R., Păun, Gh. and Pérez-Jiménez, M.J. Tissue P Systems with channel states. *Theoretical Computer Science* **330**, (2005), 101–116.
4. Frisco, P. and Hoogeboom, H.J. Simulating counter automata by P systems with symport/antiport. *Lecture Notes in Computer Science* **2597**, (2003), 288–301.
5. Garey, M.R. and Johnson, D.S. *Computers and Intractability A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, (1979).
6. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J. and Romero-Campero, F.J. A linear solution for QSAT with Membrane Creation. *Lecture Notes in Computer Science* **3850**, (2006), 241–252.
7. Ionescu, M., Păun, Gh. and Yokomori, T. Spiking neural P systems, *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308.
8. Ito, M., Martín Vide, C., and Păun, Gh. A characterization of Parikn sets of ET0L laguages in terms of P systems. In M. Ito, Gh. Păun, S. Yu (eds.) *Words, Semigroups and Transducers*, World Scientific, Singapore, 2001, 239-254.
9. Krishna, S.N., Lakshmanan K. and Rama, R. Tissue P Systems with Contextual and Rewriting Rules. *Lecture Notes in Computer Science* **2597**, (2003), 339–351.
10. Lakshmanan K. and Rama, R. On the Power of Tissue P Systems with Insertion and Deletion Rules. In A. Alhazov, C. Martín-Vide and Gh. Păun (eds.) *Preproceedings of the Workshop on Membrane Computing*, Tarragona, Report RGML 28/03, (2003), 304–318.
11. Leporati, A., Zandron, C., Ferretti, C., Mauri, G. On the computational power of spiking neural P systems. *International Journal of Unconventional Computing*, 2007, in press.
12. Maass, W., Bishop, C. eds.: *Pulsed Neural Networks*, MIT Press, Cambridge, (1999).
13. M.A. Gutiérrez–Naranjo, M.J. Pérez–Jiménez, A. Riscos–Núñez and F.J. Romero–Campero. On the power of dissolution in P systems with active membranes. *Lecture Notes in Computer Science* 3850 (2006), 224–240.
14. Martín Vide, C. Pazos, J. Păun, Gh. and Rodríguez Patón, A. A New Class of Symbolic Abstract Neural Nets: Tissue P Systems. *Lecture Notes in Computer Science* **2387**, (2002), 290–299.
15. Martín Vide, C. Pazos, J. Păun, Gh. and Rodríguez Patón, A. Tissue P systems. *Theoretical Computer Science*, **296**, (2003), 295–326.

16. Pan, L. and Ishdorj, T.-O. P systems with active membranes and separation rules. *Journal of Universal Computer Science*, **10**, 5, (2004), 630–649.
17. Păun, Gh. Computing with membranes. *Journal of Computer and System Sciences*, **61**, 1, (2000), 108–143.
18. Păun, Gh. Attacking **NP**-complete problems. In *Unconventional Models of Computation, UMC'2K* (I. Antoniou, C. Calude, M. J. Dinneen, eds.), Springer-Verlag, 2000, 94-115.
19. Păun, Gh. *Membrane Computing. An Introduction*. Springer–Verlag, Berlin, (2002).
20. Păun, A. and Păun, Gh. The power of communication: P systems with symport/antiport. *New Generation Computing*, **20**, 3, (2002), 295–305.
21. Păun, Gh. and Pérez-Jiménez, M.J. Recent computing models inspired from biology: DNA and membrane computing. *Theoria*, **18**, 46, (2003), 72–84.
22. Păun, Gh., Pérez-Jiménez, M.J. and Riscos-Núñez, A. Tissue P System with cell division. *In. J. of Computers, communications & control*, **3**, 3, (2008), 295–303.
23. Gh. Păun, Y. Sakakibara, T. Yokomori: P systems on graphs of restricted forms. *Publicationes Mathematicae Debrecen*, 60 (2002), 635–660.
24. Pérez-Jiménez, M.J., Romero-Jiménez, A. and Sancho-Caparrini, F. The P versus NP problem through cellular computing with membranes. *Lecture Notes in Computer Science* **2950**, (2004), 338–352.
25. Pérez-Jiménez, M.J., Romero-Jiménez, A. and Sancho-Caparrini, F. A polynomial complexity class in P systems using membrane division. *Journal of Automata, Languages and Combinatorics*, **11**, 4, (2006), 423-434.
26. Prakash, V.J. On the Power of Tissue P Systems Working in the Maximal-One Mode. In A. Alhazov, C. Martín-Vide and Gh. Păun (eds.). *Preproceedings of the Workshop on Membrane Computing*, Tarragona, Report RGML 28/03, (2003), 356–364.
27. Zandron, C., Ferretti, C. and Mauri, G. Solving NP-Complete Problems Using P Systems with Active Membranes. In I. Antoniou, C.S. Calude and M.J. Dinneen (eds.). *Unconventional Models of Computation, UMC'2K*, Springer-Verlag, (2000), 289–301.
28. ISI web page `http://esi-topics.com/erf/october2003.html`
29. P systems web page `http://ppage.psystems.eu/`